PROVIDING MULTICAST VIDEO ON DEMAND USING NATIVE-MODE ASYNCHRONOUS TRANSFER MODE

J. W. Lockwood, S. M. Kang, A. Hossain, J. Hiltenbrant

University of Illinois
Department of Electrical and Computer Engineering
405 North Mathews Ave
Urbana, IL 61801
ipoint@ipoint.vlsi.uiuc.edu

ABSTRACT

A network video server and client devices have been designed and prototyped that use multicast to deliver Motion-JPEG, MPEG-1, and MPEG-2 video on top of IP, IP-over-ATM and native-mode ATM (AAL5) protocols. The server runs on a workstation-class machine, while the clients have been implemented both as a multi-threaded software entity and as a stand-alone ATM network-attached playback device. Scalability of the video distribution network is obtained by using multicast features of the underlying network. Feedback from the clients and a single video frame retransmission mechanism are used to decrease end-to-end frame loss.

1. INTRODUCTION

Multicast networks are well suited for delivery of video. Rather than transmitting duplicate frames to multiple clients, a video source sends one frame and lets the duplication of data occur in a distributed nature throughout the switches of the network.

Asynchronous Transfer Mode networks have great utility for the delivery of compressed video. They can transport variable-bit-rate data while enforcing minimum bandwidth and delay requirements.

The video server can be inexpensively implemented using a network-attached workstation. Such a server encapsulates MPEG data into ATM Adapation Layer 5 (AAL5) frames and then transmits these frames over the network. The video client can be implemented either as software entity on a network-attached workstation or as a standalone network device.

For video streaming, endpoint clients must buffer data. At minimum, the client must buffer an amount of stored video data proportional to the network jitter. Storing an amount of video proportional to the round-trip delay between the client and server, however, can more dramatically increase the quality and scalability of the video distribution network.

This research has been supported by National Science Foundation Engineering Research Center grant ECD 89-43166 and Advance Research orogram Agency (ARPA) grant for Center for Optoelectronic Science and Technology (COST) grant MDA 972-94-1-0004

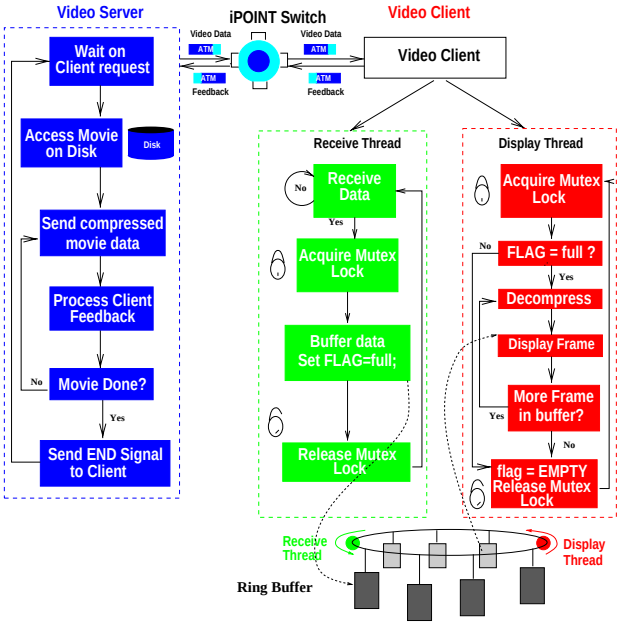

Figure 1: Video Server and Client Software Architecture.

2. VIDEO SERVER

The iPOINT video server is an application program that accesses a database of Motion-JPEG (MJPEG), MPEG-1, and MPEG-2 video clips and transmits their contents to remote network clients. The server makes use of underlying multicast features of an IP or native-mode ATM network. The software architecture of the server is shown on the lefthand

side of Figure 1.

The server process begins by listening at a well-known address for an incoming client request. Upon receiving an initial request, the server establishes a multicast connection to the client and accepts a unicast connection in the reverse path. Subsequent clients join the conference by adding themselves to the multicast group, and by each client establishing one additional unicast connection back to the server.

Once the connections have been established, the server `mmaps` the file to memory and gradually reads video frames from disk [1]. These video frames are divided into chunks, and each chunk is sent individually. Internet traffic is transmitted using IP multicast datagrams, while native-mode ATM traffic is sent using AAL5 frames over a multipoint virtual circuit. The size of a chunk can be a variable number of bytes, but is limited to the size set by the `setsocketopt()` call for IP and to 9180 bytes for AAL5 frames (the MTU).

The server uses feedback from the clients to control the transmission of the video frames. The server temporarily buffers recently sent frames in memory for possible *retransmit* requests. The server gives high priority for retransmission of these chunks, and will send them before other frames that have already been scheduled for transmission.

3. VIDEO CLIENT

The video client is implemented as a multithreaded process. The *receive* thread reads data from the network and stores it to a ring buffer. The *display* thread decompresses data from the ring buffer and displays it to the screen. Together, the *receive* and *display* threads chase each other around the ring buffer. The software architecture of the video client is shown on the righthand side of Figure 1.

The client is responsible for detecting lost chunks of a transmitted video frame. For each video frame, the client knows exactly how many chunks to receive from information contained in the packet header sent by the server. The client identifies any delayed, duplicated, or reordered chunks then *stitches* them together into the appropriate location of the ring buffer. If the client receives an incomplete frame, it sends a *retransmit* request to the server with a bit array indicating which chunks of this video frame need to be retransmitted. Sending the bit array sequence requires less CPU and network resources than sending an independent retransmit request for each lost chunk. Out-of-order frames are not a problem for ATM.

4. HANDLING MULTICAST FEEDBACK

Handling feedback from members of a multicast group on the server's end is important. An implosion of feedback messages on the server must be avoided as the multicast group grows in size. We avoid this feedback implosion

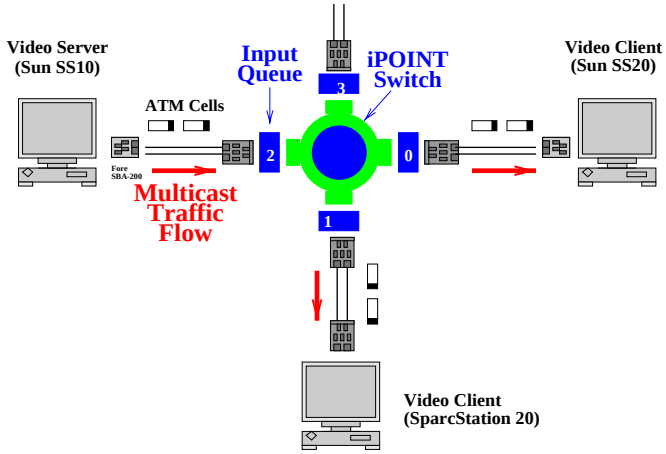

Figure 2: Video Testbed Setup.

on the server by implementing *probabilistic suppression*, which is derived from the implementation of `wb` [2] [3].

When any one of the members of the multicast client group detect a packet loss, the client sets up a *request timer*. The initial value of this timer is chosen from the uniform distribution $[0, T_{expire}]$, where T_{expire} indicates the final time at which an incomplete frame repair request can still be sent to the server before the frame is decompressed. It is a function of the distance (in units of ring-node elements) between the *receive* and *display* threads, and the rate at which data is consumed from these elements by the client.

When the server receives a repair request, it retransmits the lost chunk on the multicast channel. Doing so allows all clients to receive the repair nearly at once. When a client receives a repair for a lost chunk, it cancels any pending request timers for that data. If the client needs the chunk, it will update data in the ring buffer. Other clients will discard the data since it had already been received correctly.

5. IPOINT TESTBED

Evaluation and measurement of the video server and clients were performed using the iPOINT testbed [4]. Workstations equipped with Fore SBA-200 ATM host adapter interfaces were used to transmit AAL5 frames. The ATM cells generated from these cards were carried by full-duplex, 100Mbps fiber links to an input queue of the iPOINT switch [5]. The switch then used atomic multicast to simultaneously transmit incoming cells to multiple output ports of the switch [6]. A diagram of the network topology is shown in Figure 2.

6. PERFORMANCE MEASUREMENTS

The throughput of the video server was measured in the testbed for increasing numbers of server threads. Each thread

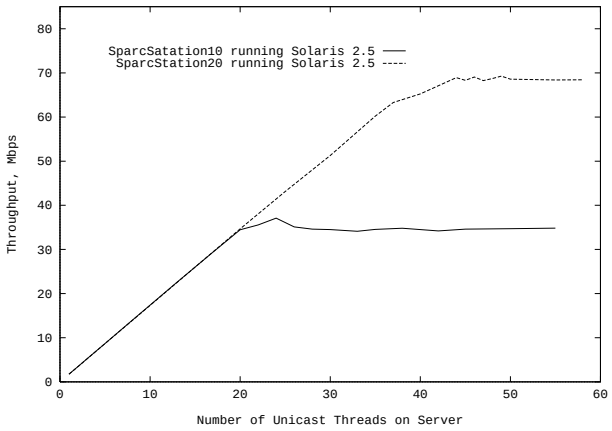

Figure 3: Video Server Performance.

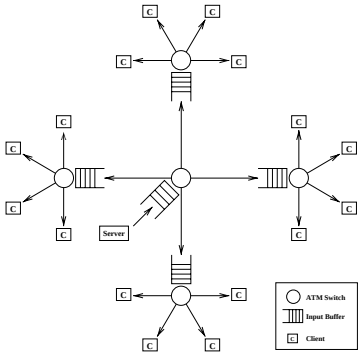

Figure 4: Multicast Distribution Tree.

transmitted a real MJPEG stream with an average bandwidth of 1.4 Mbps. Results of the server performance for both a Sun SparcStation 20 (SS20) and for a Sun SparcStation 10 (SS10) are shown in Figure 3. The throughput of the server scaled almost linearly with the number of threads up until the point where the the CPU utilization saturated at 100%.

Performance results for a large number of clients on a multicast network were determined by combining the measured results for the single server experiment with a probabilistic analysis of loss through multiple switches. For this analysis, we considered input buffered ATM switches implementing atomic multicast with a fanout degree of 4 per switch. Analysis was performed for 1, 2 and 3 levels of ATM switches connected in a star topology. Figure 4 shows such a network with 2 levels of ATM switches.

The mathematical formulation for the experiment are given below:

$$\begin{aligned} P_{drop} &= \text{Probability of dropping chunk} \\ P_{retran} &= \text{Probability of chunk retransmission} \\ P_{lost} &= \text{Probability of lost chuck} \end{aligned}$$

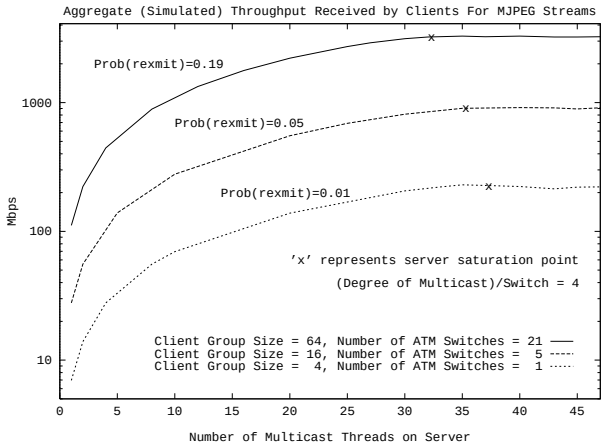

Figure 5: Aggregate Video Server Throughput.

$$\begin{aligned} M &= \text{Degree of multicast per switch} \\ L &= \text{Depth of multicast tree} \\ N &= \text{Total number of ATM switches} \\ C_m &= \text{Multicast group size (Clients)} \end{aligned}$$

Using the above definitions, the following holds true for the multicast group in Figure 4:

$$N = 1 + M + M^2 + M^3 + \cdots \quad (1)$$

$$= \frac{1 - M^L}{1 - M},\ M \geq 2 \quad (2)$$

$$C_m = M^L \quad (3)$$

$$P_{retran} = 1 - (1 - P_{drop})^N \quad (4)$$

$$P_{lost} = 1 - P_{retran}^2 \quad (5)$$

Setting P_{drop} to 0.01 (1.0%) reflects a reasonable frame loss due to bit errors and queue overflows. Setting $M = 4$ models a typical multicast fanout. Figure 5 shows the aggregate throughput received by all the clients which are participating in this multicast reception. From the above equations, with $L = 3$, we have, $C_m = 64$, $N = 21$, $P_{retran} = 0.19$. Under these conditions, the aggregate throughput received by the clients saturates near $3\ Gbps$ with just over 30 threads and a lost frame rate of under 4%.

Similarly, for $L = 2$ and $L = 1$, we find $P_{retran} = 0.05$ and 0.01, respectively, for the same value of $P_{drop} = 1.0\%$.

Allowing any client to trigger a retransmit works especially well for ATM. QoS parameters distribute cell loss evenly among clients. Mechanisms to handle asymmetric conditions with more heterogeneous groups is the topic of [7].

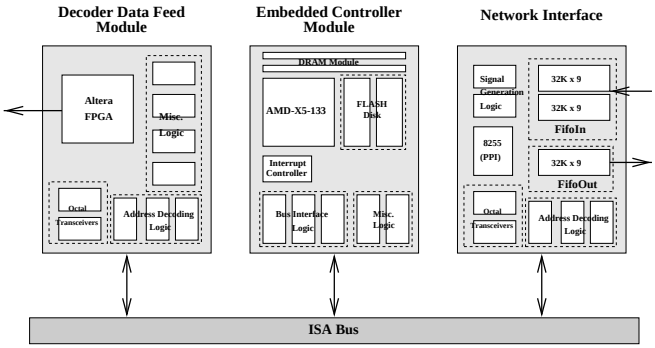

Figure 6: iVIM Module.

7. HARDWARE CLIENT

Using the architecture developed for the software-based video client, we have developed a standalone MPEG-2 playback client called the Illinois Video Interface Module (iVIM). This network device directly receives MPEG-2 frames using AAL5 over an ATM link and decodes them to full-motion video without the assistance of a workstation.

The components of the system include a field programmable gate array *decoder data feed module* [8], a 5x86 embedded controller, and a commercially-available MPEG-2 decoding chipset [9]. A block diagram of the system is shown in Figure 6. Video frames enter the system from the right as AAL5 frames. The cells are passed across an ISA interface to the embedded controller (middle). The embedded controller, in turn, manages the ring buffer and controls feedback to the server. The data-feed module (left) feeds the MPEG-2 decoder with a continous stream of MPEG-2 frames.

8. CONCLUSION

A video server and clients have been developed that provided effective transport of MJPEG, MPEG-1, and MPEG-2 frames using IP, IP-over-ATM, and native-mode ATM. Multicast was used to enable transmission of video to a large numbers of clients. The use of a single video frame retransmission mechanism dramatically increased the probability of frames being received correctly. Measurement of the server performance and analysis of probabilistic network loss demonstrated the feasibility of a workstation-class server feeding 3 Gbps of aggregate traffic using 32 independent MJPEG video streams.

9. REFERENCES

[1] B. Smith, “Implementation techniques for continuous media systems and applications.” PhD Thesis, Computer Science, UC Berkeley, 1994.

[2] S. Floyd, V. Jacobson, and S. McCanne, “A reliable multicast framework for light-weight sessions and application level framing,” in *Proceedings of SIGCOMM 95*, pp. 342–356, 1995.

[3] V. Jacobson, “A portable public domain network whiteboard.” Xerox PARC Viewgraphs, Apr. 1992.

[4] J. W. Lockwood, H. Duan, J. J. Morikuni, S. M. Kang, S. Akkineni, and R. H. Campbell, “Scalable optoelectronic ATM networks: The iPOINT fully functional testbed,” *IEEE Journal of Lightwave Technology*, pp. 1093–1103, June 1995.

[5] H. Duan, J. W. Lockwood, S. M. Kang, and J. Will, “High-performance oc-12/oc-48 queue design prototype for input-buffered ATM switches,” in *IEEE Infocom*, (Kobe, Japan), pp. 20–28, Apr. 1997.

[6] J. W. Lockwood, S. M. Kang, S. G. Bishop, H. Duan, and A. Hossain, “Development of the iPOINT testbed for optoelectronic asynchronous transfer mode networking,” in *International Conference on Information Infrastructure*, pp. 509–513, Apr. 1996.

[7] T. Turletti and I. Wakeman, “Scalable feedback control for multicast video distribution in the internet,” in *Proceedings of SIGCOMM 94*, pp. 58–67, 1994.

[8] D. Taubert, “Design and implementation of a hardware MPEG-2 decoder system.” MS Thesis, Computer Science, University of Illinois, 1996.

[9] “Hdm8211p: MPEG-2 systems, audio and video decoder hardware.” Odeum Microsystems Inc, 1996.